

Provenance for Reproducible Data Science

Andreas Schreiber

German Aerospace Center (DLR)
Cologne/Berlin, Germany

PyData Seattle 2017



Knowledge for Tomorrow



Topics

- Introduction to provenance and PROV
- Modelling provenance for data processing
- Python APIs for provenance recording
- Provenance recording for Jupyter notebooks
- Storing provenance in graph databases
- Analysis of provenance information



Introduction

Study Industrial Mathematics
(*dt. Technomathematik*)



Soldier (SIGINT)



Scientist (Grid Computing)



Introduction



**Deutsches Zentrum
für Luft- und Raumfahrt**
German Aerospace Center

Simulation and Software Technology, Cologne/Berlin
Head of Intelligent and Distributed Systems department

Institute of Data Science, Jena
Head of Secure Software Engineering group



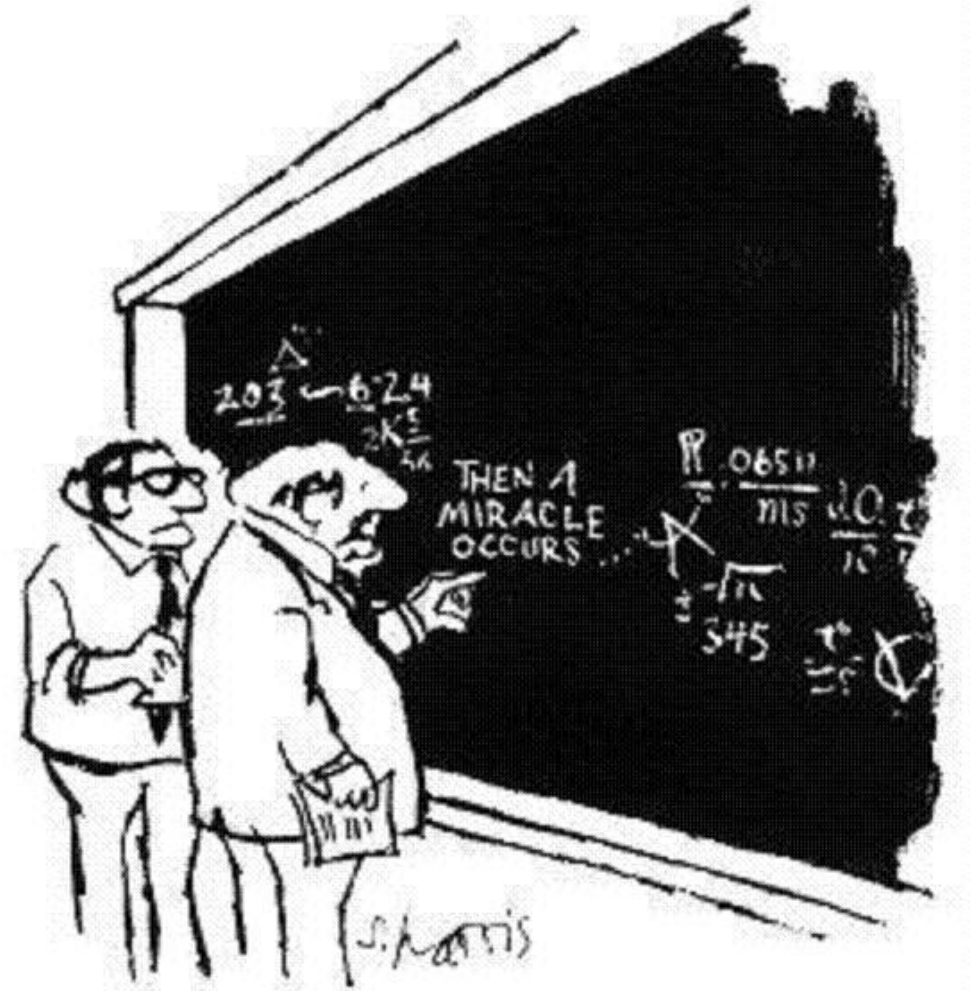
**Co-Founder
Data Scientist
Patient**



Reproducibility

Reproducibility *in (data) science* is based on

- Open Source Software
- Code Reviews
- Code Repositories
- Publications with code
- Container (Docker etc.)
- Workflows
- (Electronic) laboratory notebooks
- Open data formats
- Data management
- **Metadata and Provenance**



"I think you should be more explicit here in step two."

Metadata and Provenance

Each data file in data processing has two parts

- **Data:** Actual measured, simulated, or generated data results
- **Metadata:** Information that describes or relates to the data

Metadata can include a large variety of information

- Geographic information that can be used to limit a spatial search
- Quality information (*“Data is bad for some reason,” “Granule is cloud obscured”*)
- Instrument configuration information (*“Instrument in spectral zoom mode,” “Spacecraft maneuver in progress”*)
- Extra information about the data files themselves: file size, checksum for data integrity verification
- Provenance information (*Where did I get this file? How did it come to exist?*)



Provenance

Basics

- Provenance refers to the source of information and the process that led to its existence
- Provenance information is critical to users trying to understand where a particular data file came from

Other and related terms

- Traceability
- Lineage
- Logging
- Monitoring



Provenance Information

***Capture, archive, and distribute* provenance information, for example**

- The source of all externally supplied data files
- The source of the algorithms used to transform the data within the system
- Algorithm Design Documents
- A complete description of the processing environment
- A complete description of the processing framework
- A record of each job's execution



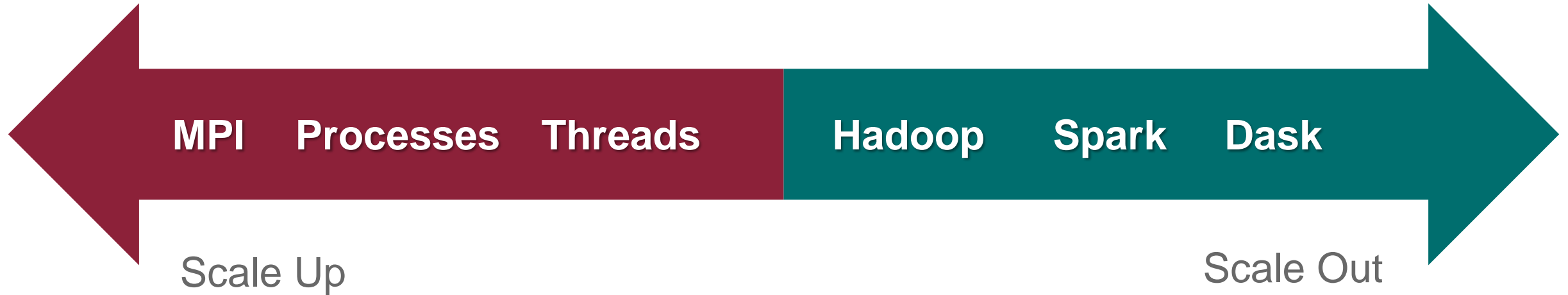
Big Data Processing

High-Performance Computing

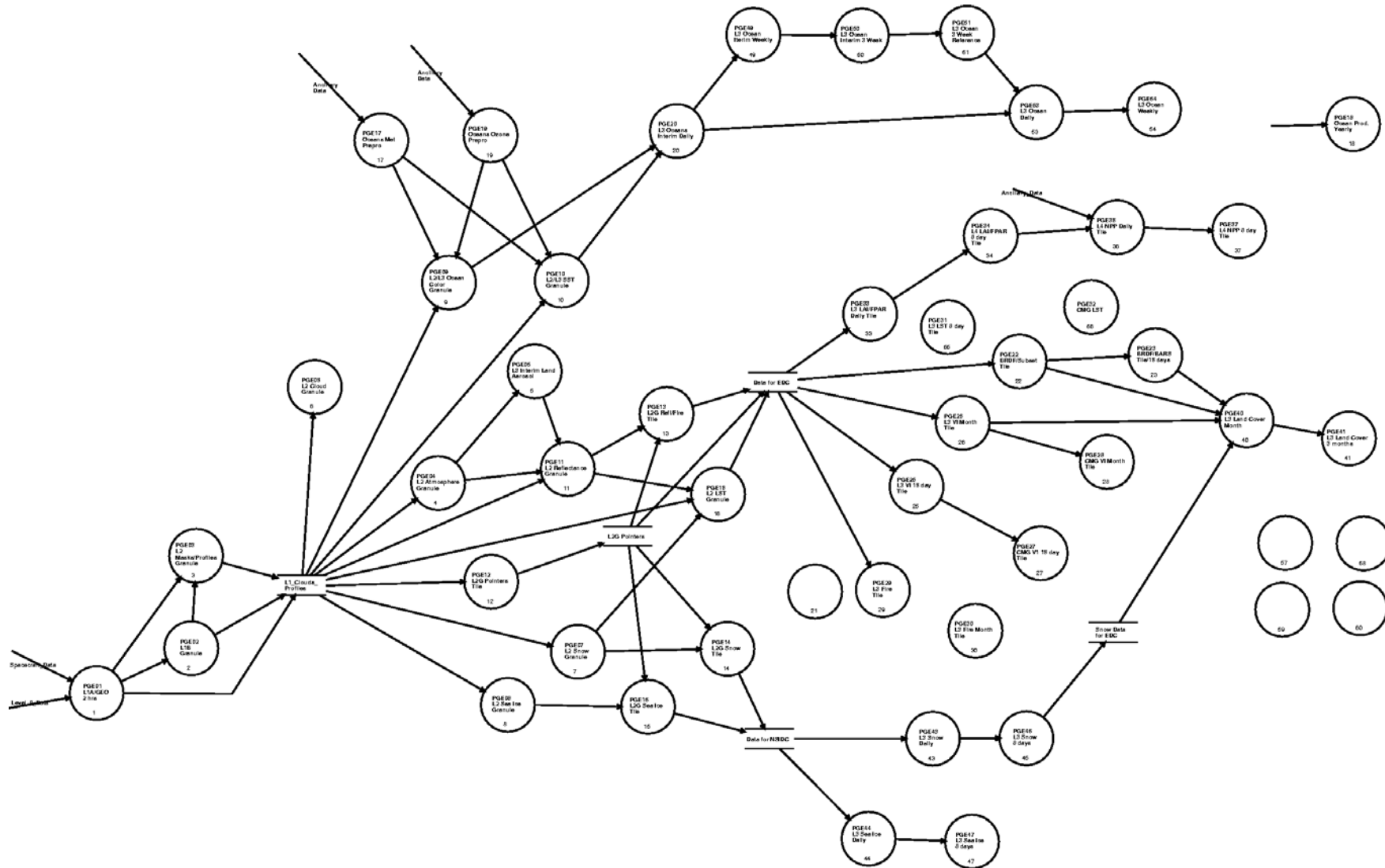
- Explicit Control

Distributed Computing

- Implicit Control (via Graphs)



Data Science Workflows



More Formal Definition of Provenance



Provenance is

information about entities, activities, and people
involved in
producing a piece of data or thing,
which can be used to form
assessments about its quality, reliability or trustworthiness.

PROV W3C Working Group
<https://www.w3.org/TR/prov-overview>



W3C Specification „PROV“



- **PROV-O**, the PROV ontology, an OWL2 ontology allowing the mapping of the PROV data model to RDF
- **PROV-DM**, the PROV data model for provenance
- **PROV-N**, a notation for provenance aimed at human consumption
- **PROV-CONSTRAINTS**, a set of constraints applying to the PROV data model
- **PROV-XML**, an XML schema for the PROV data model
- **PROV-AQ**, mechanisms for accessing and querying provenance
- **PROV-DICTIONARY** introduces a specific type of collection, consisting of key-entity pairs
- **PROV-DC** provides a mapping between PROV-O and Dublin Core Terms
- **PROV-SEM**, a declarative specification in terms of first-order logic of the PROV data model
- **PROV-LINKS** introduces a mechanism to link across bundles



PROV Elements



Entities

- Physical, digital, conceptual, or other kinds of things
- For example, documents, web sites, graphics, or data sets

Activities

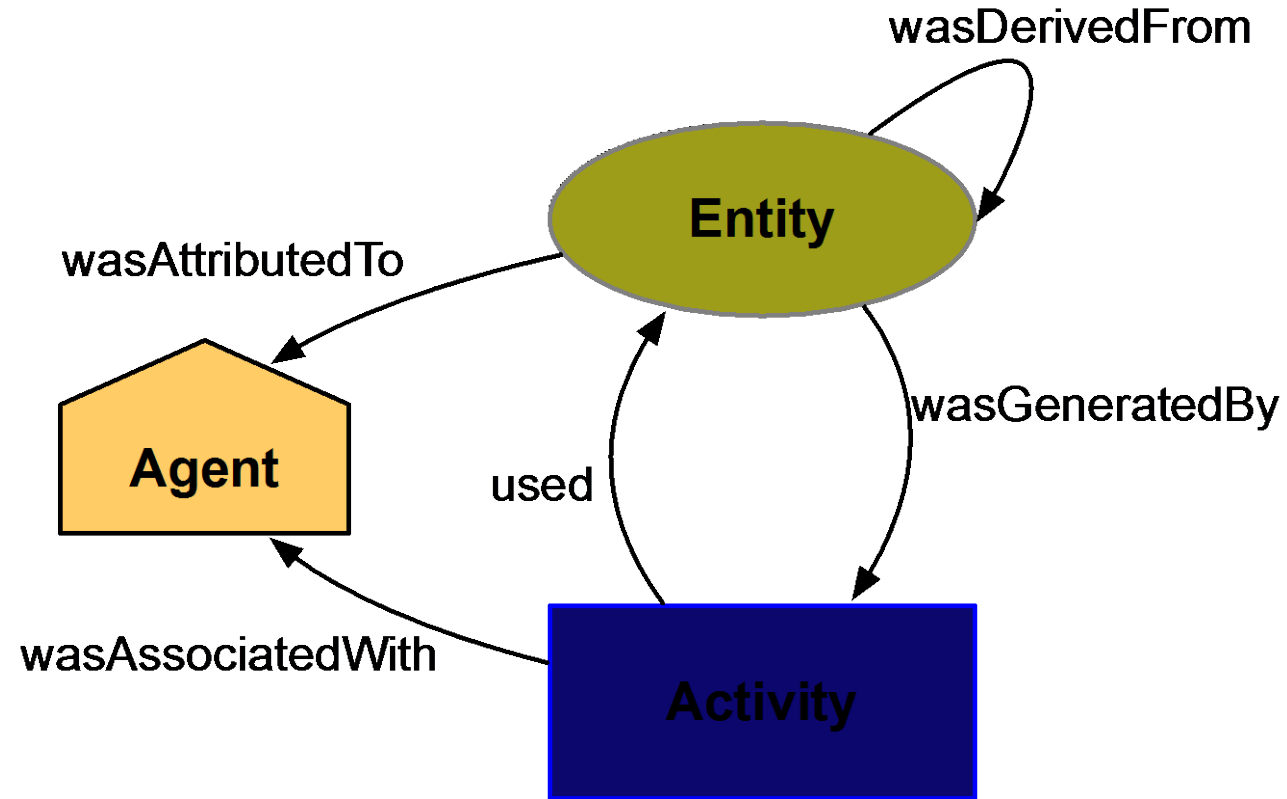
- Activities *generate* new entities or make *use* of existing entities
- Activities could be actions or processes

Agents

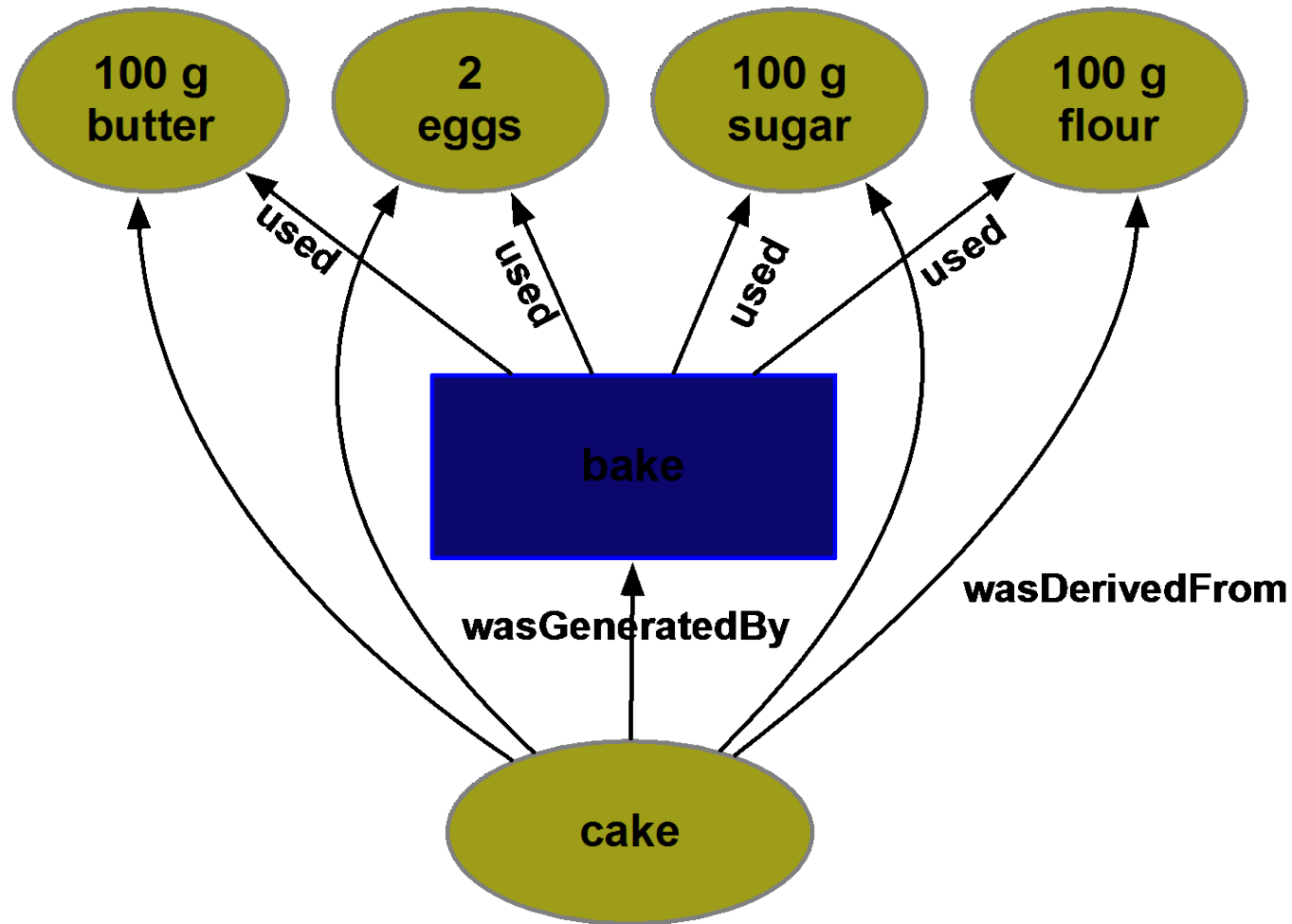
- Agents takes a role in an activity and have the responsibility for the activity
- For example, persons, pieces of software, or organizations



PROV Relations



Baking a Cake



PROV Notations and Representations

Textual Representations

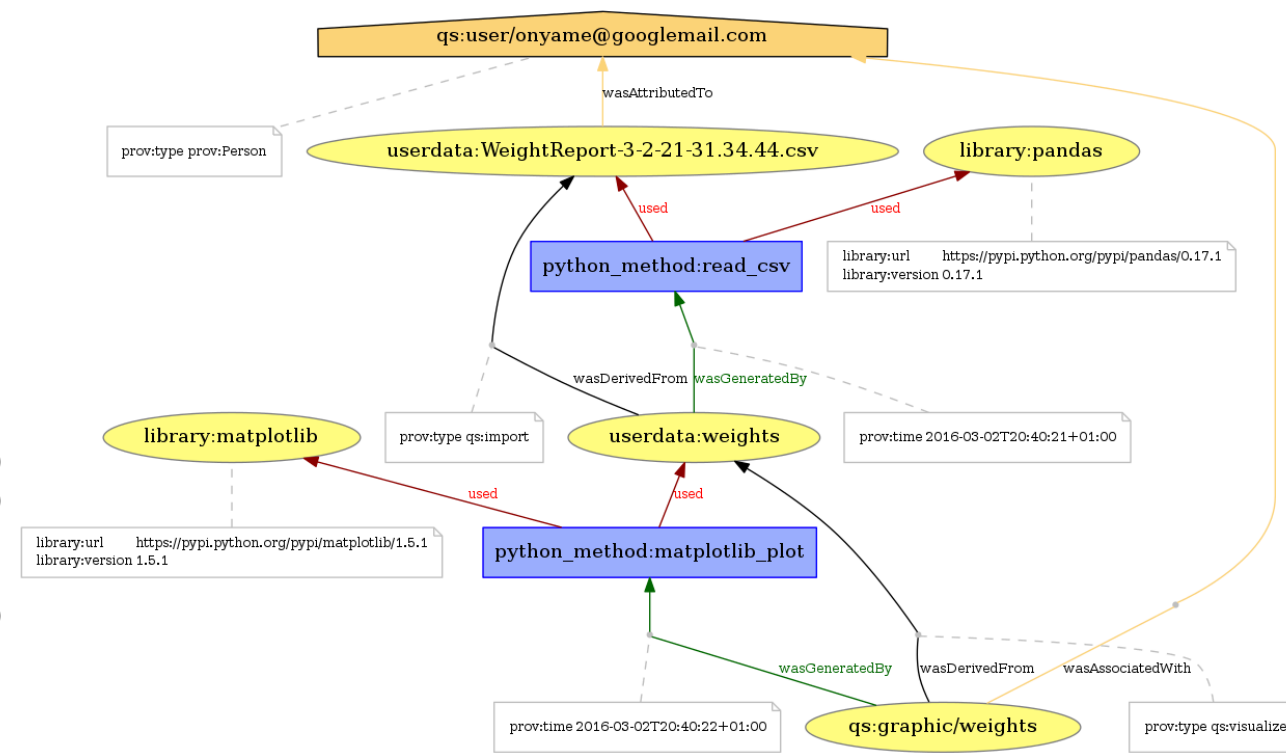
- Formats: *PROV-N*, *JSON*, *Turtle*, *XML*, ...

```
document
  prefix userdata http://software.dlr.de/qs/userdata/
  . . .
  wasDerivedFrom(userdata:weights,
userdata:WeightReport.csv,
  wasDerivedFrom(qs:graphic/weights, userdata:weights,
  wasAssociatedWith(qs:graphic/weights,
qs:user/onyame@gmail.com, -)
  used(python_method:read_csv, library:pandas, -)
  used(python_method:matplotlib_plot, userdata:weights, -)
  used(python_method:matplotlib_plot, library:matplotlib, -)
  used(python_method:read_csv, userdata:WeightReport.csv, -)
  wasAttributedTo(userdata:WeightReport.csv,
qs:user/onyame@gmail.com)
  agent(qs:user/onyame@gmail.com, [prov:type="prov:Person"])
  entity(library:pandas, [library:version="0.17.1"])
  entity(userdata:WeightReport.csv)
  entity(userdata:weights)
```

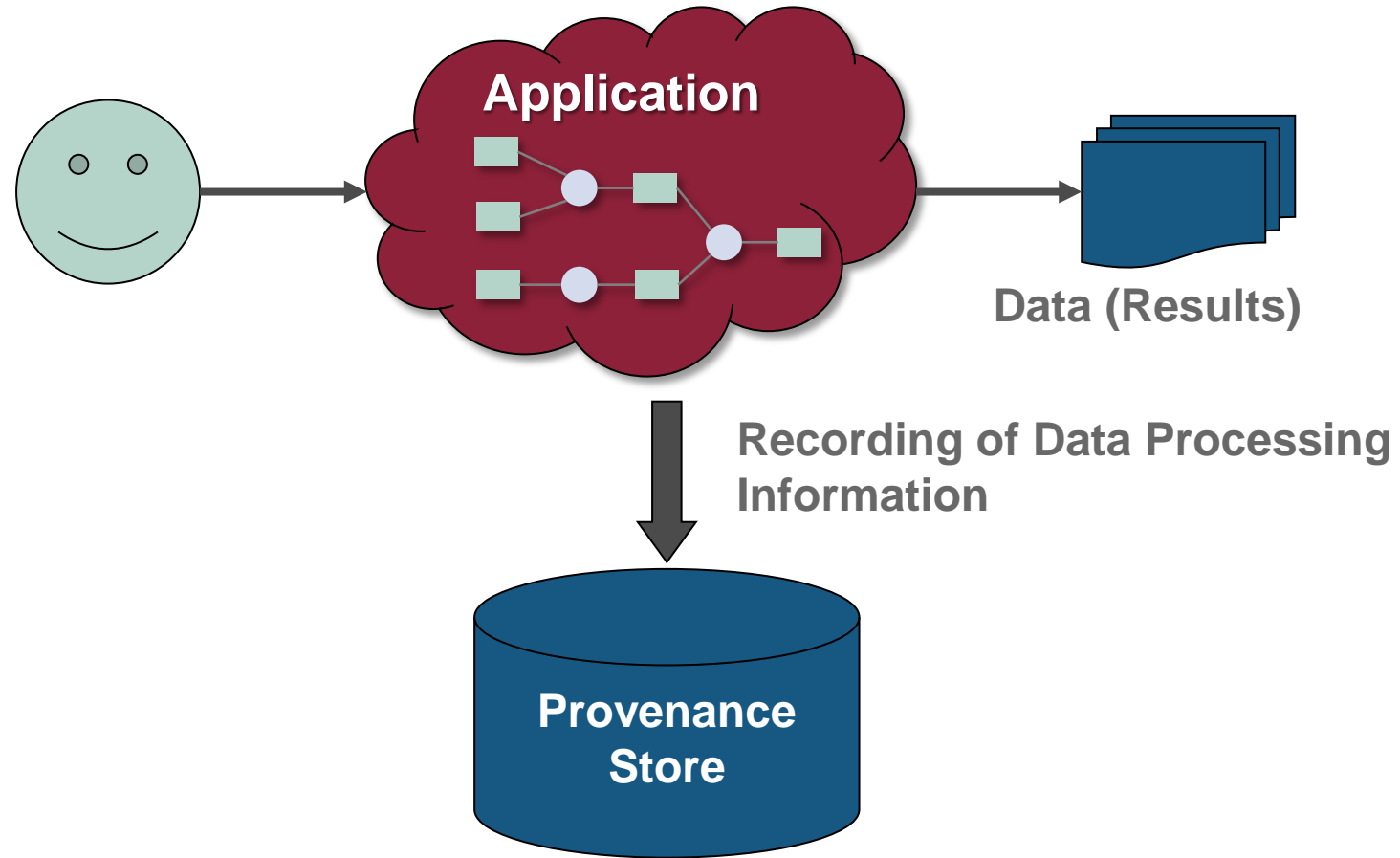
. . .
endDocument



Visualizations



Provenance Architecture



Storing and Retrieving Provenance

Some Storage Technologies

- Relational databases and SQL
- XML and Xpath
- RDF and SPARQL
- Graph databases and Gremlin/Cypher

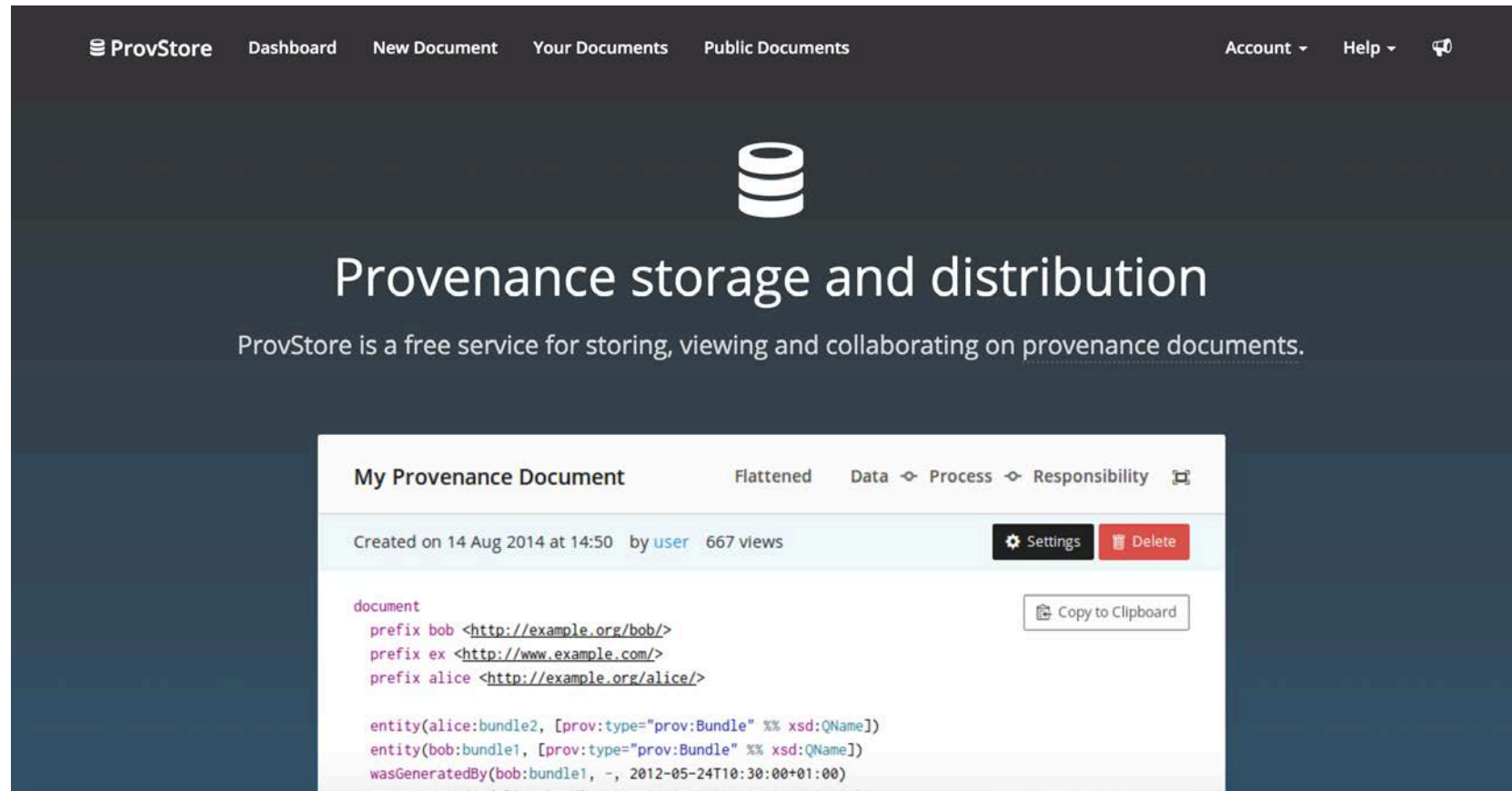
Services

- REST APIs
- PROVSTORE



ProvStore

University of Southampton

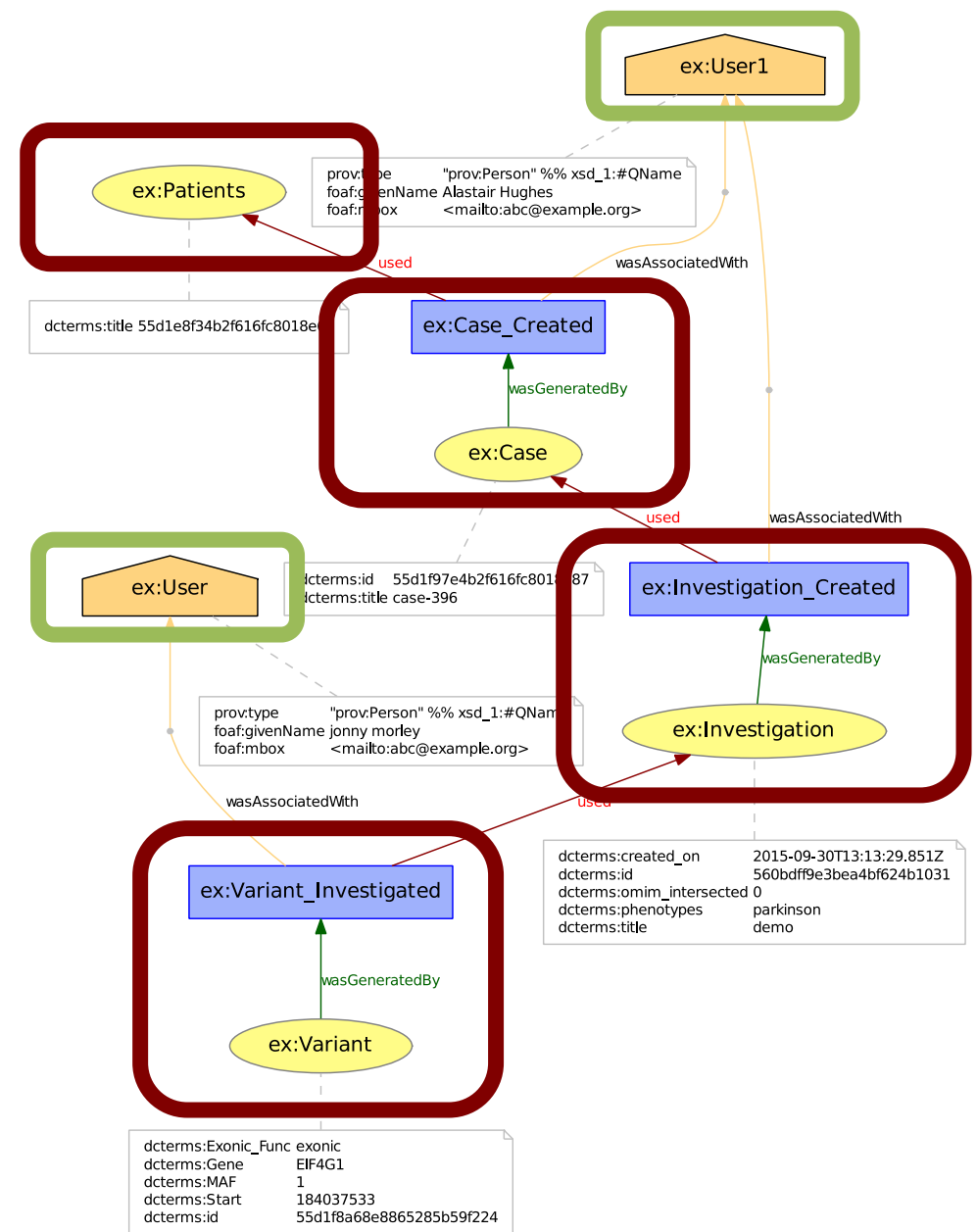
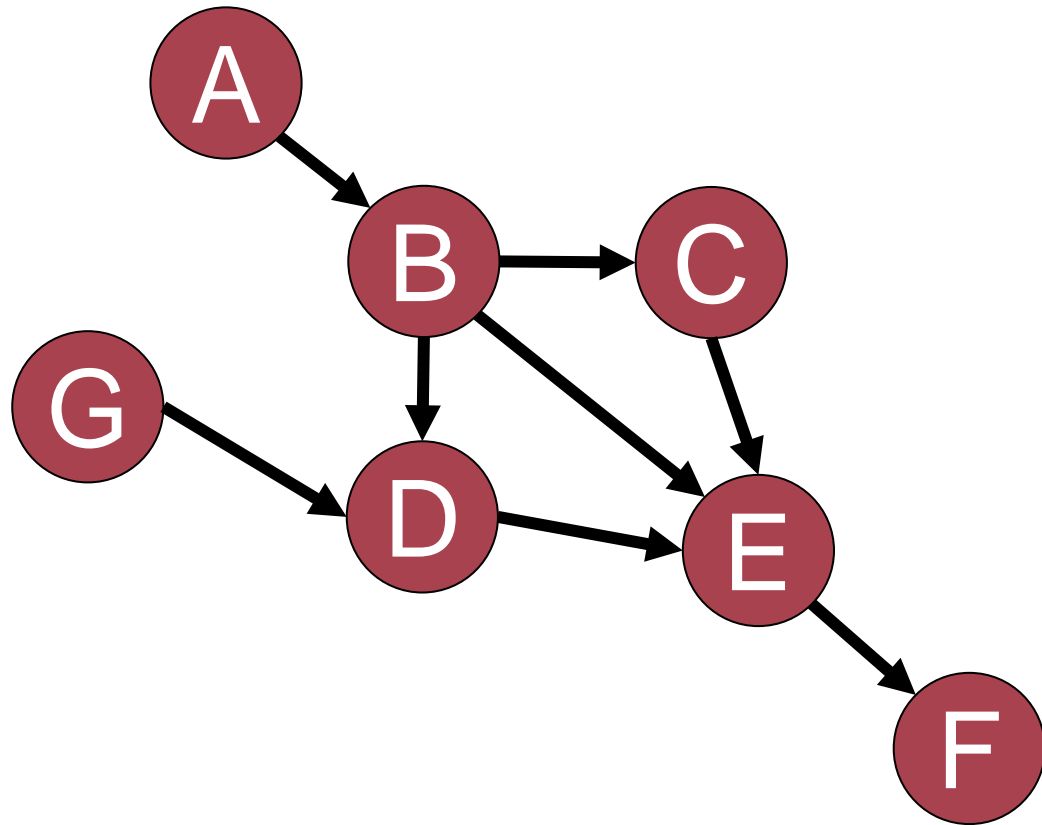


- RESTful web service
- storage and access of provenance documents
- Public and private documents
- Conversion to various text formats
- Simple visualizations
- APIs
 - Python
 - jQuery

<https://provenance.ecs.soton.ac.uk/store/>

Graphs

Provenance is a *Directed Acyclic Graph (DAG)*



Graph Databases

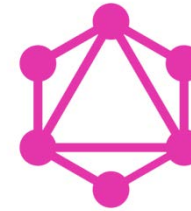
Naturally, graph databases are a good technology for storing (Provenance) graphs

Many graph databases are available

- Neo4J
- Titan
- ArangoDB
- ...

Query languages

- Cypher
- Gremlin (TinkerPop)
- GraphQL

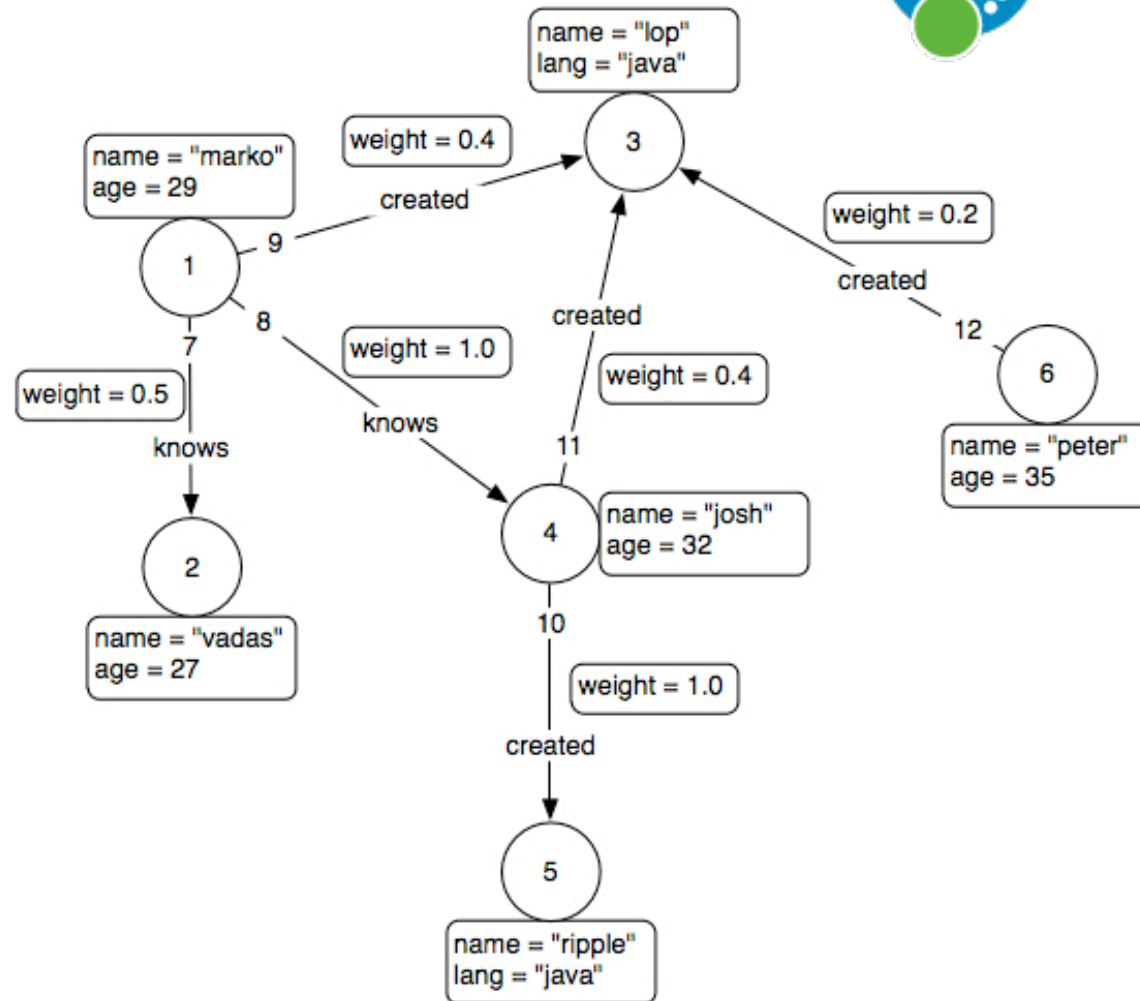


Neo4j

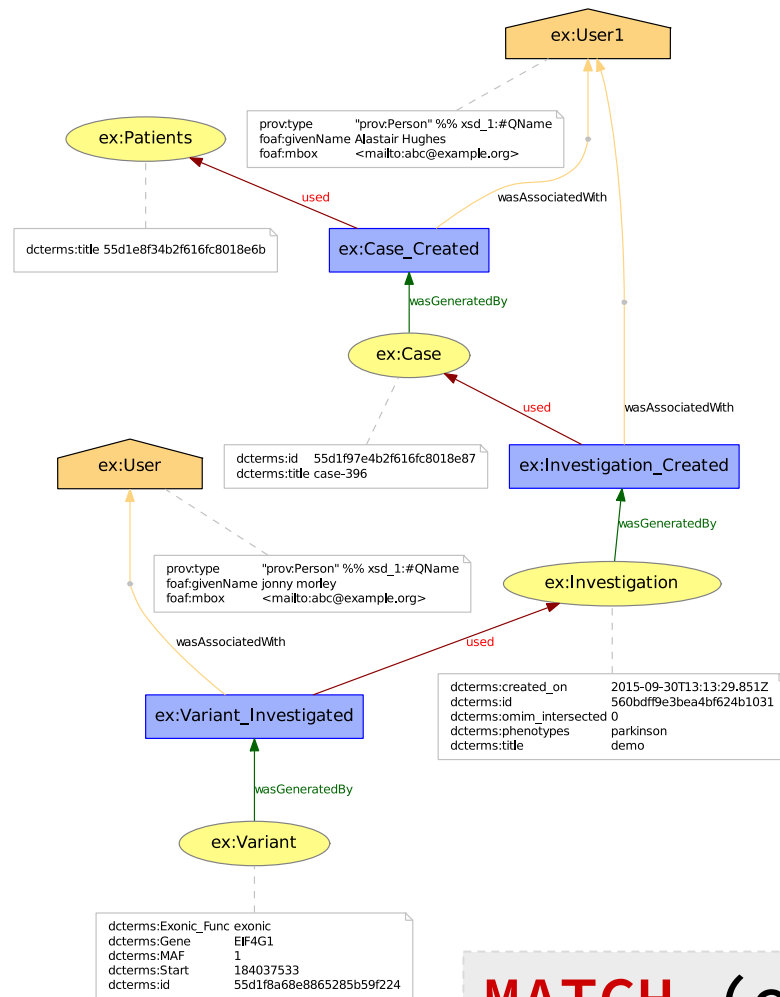


- Open-Source
- Implemented in Java
- Stores *property graphs* (key-value-based, directed)

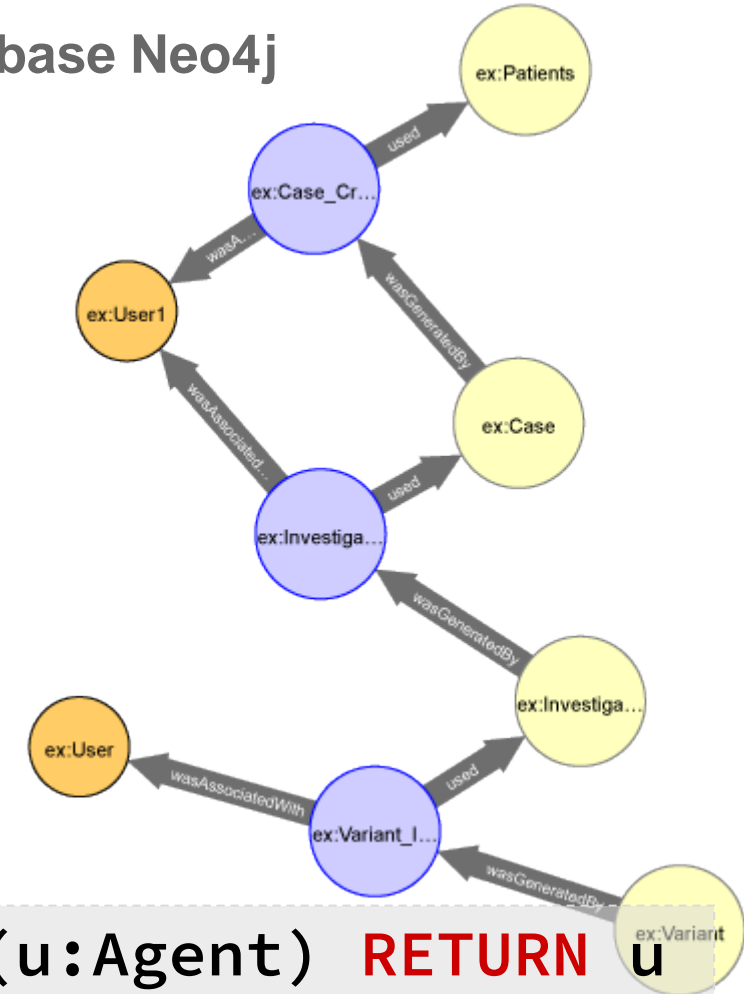
<http://neo4j.com>



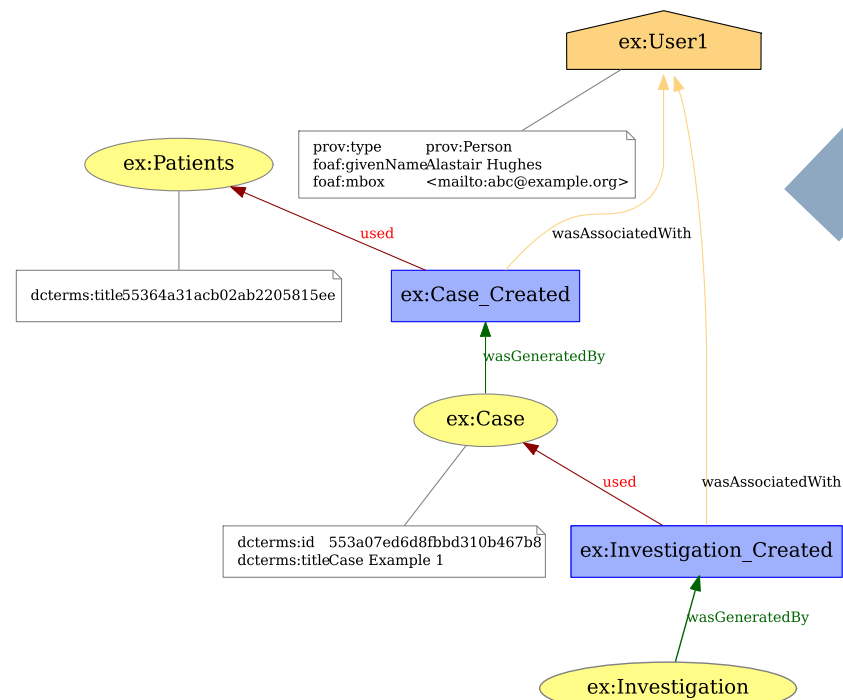
Storing Provenance in Graph Database



Graph database Neo4j



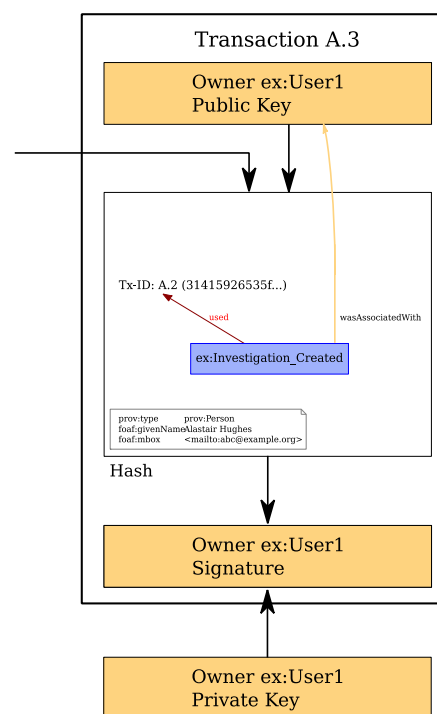
Trusted Provenance: Storing Provenance in a Blockchain



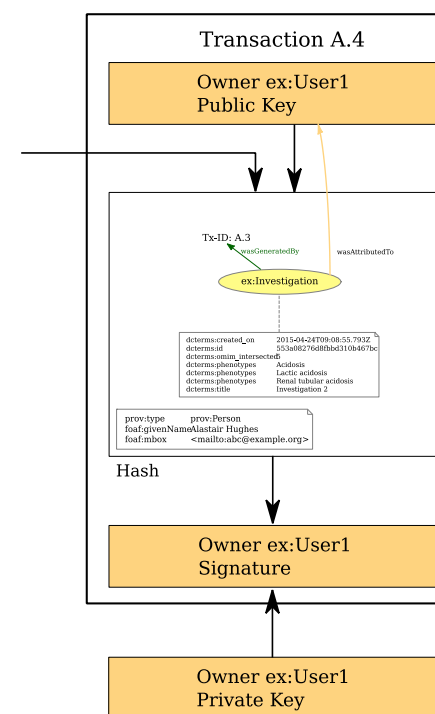
Document A

PROV2BIGCHAINDB

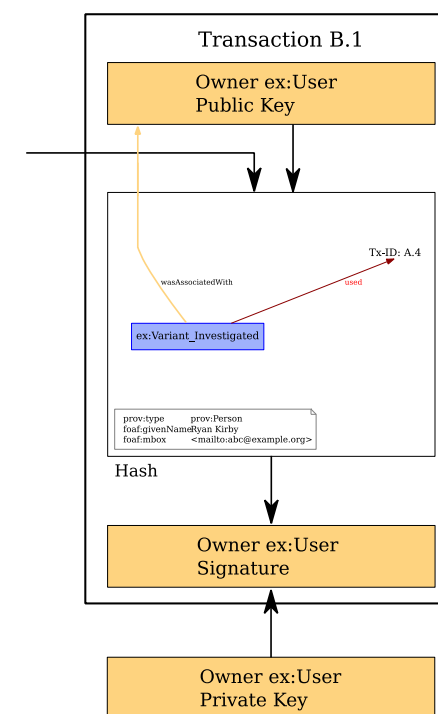
<https://github.com/DLR-SC/prov2bigchaindb>



Create Asset



Create Asset



Create Asset

Gather or Generate Provenance

Depends on your application (tools, languages, etc.)

- Generation at *run-time*, *compile-time*, or *retrospectively*

Runtime

- Instrumentation of the application
- Cumbersome from software engineering perspective
- Combined with logging or with aspect-oriented approaches

Compile time

- Based on static code analysis (dependency analysis, program slicing, etc.)

Retrospectively

- Reconstructed from files or filesystem metadata



Tools and Libraries for Python

Libraries for Python

- PROVPy
- PROVNEO4J
- PROV-DB-CONNECTOR

Other Tools

- NoWorkflow
- Git2Prov



Python Library ProvPy (PROV)

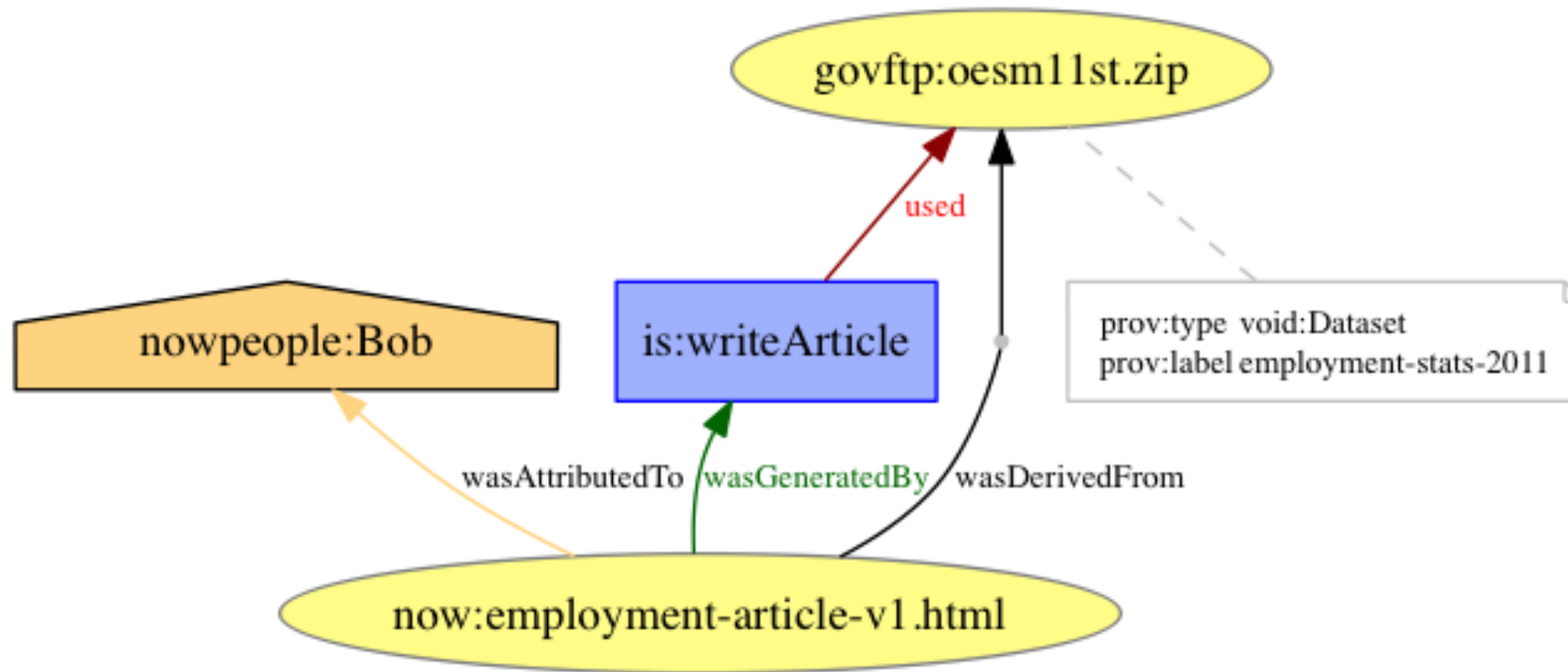
<https://github.com/trungdong/prov>

```
from prov.model import ProvDocument
# Create a new provenance document
d1 = ProvDocument()
# Entity: now:employment-article-v1.html
e1 = d1.entity('now:employment-article-v1.html')
# Agent: nowpeople:Bob
d1.agent('nowpeople:Bob')
# Attributing the article to the agent
d1.wasAttributedTo(e1, 'nowpeople:Bob')
d1.entity('govftp:oesm11st.zip',
          {'prov:label': 'employment-stats-2011',
           'prov:type': 'void:Dataset'})
d1.wasDerivedFrom('now:employment-article-v1.html',
                  'govftp:oesm11st.zip')
# Adding an activity
d1.activity('is:writeArticle')
d1.used('is:writeArticle', 'govftp:oesm11st.zip')
d1.wasGeneratedBy('now:employment-article-v1.html', 'is:writeArticle')
```



Python Library ProvPy (PROV)

<https://github.com/trungdong/prov>



PROVNEO4J – Storing PROV Documents in Neo4j

<https://github.com/DLR-SC/provneo4j>

```
import provneo4j.api

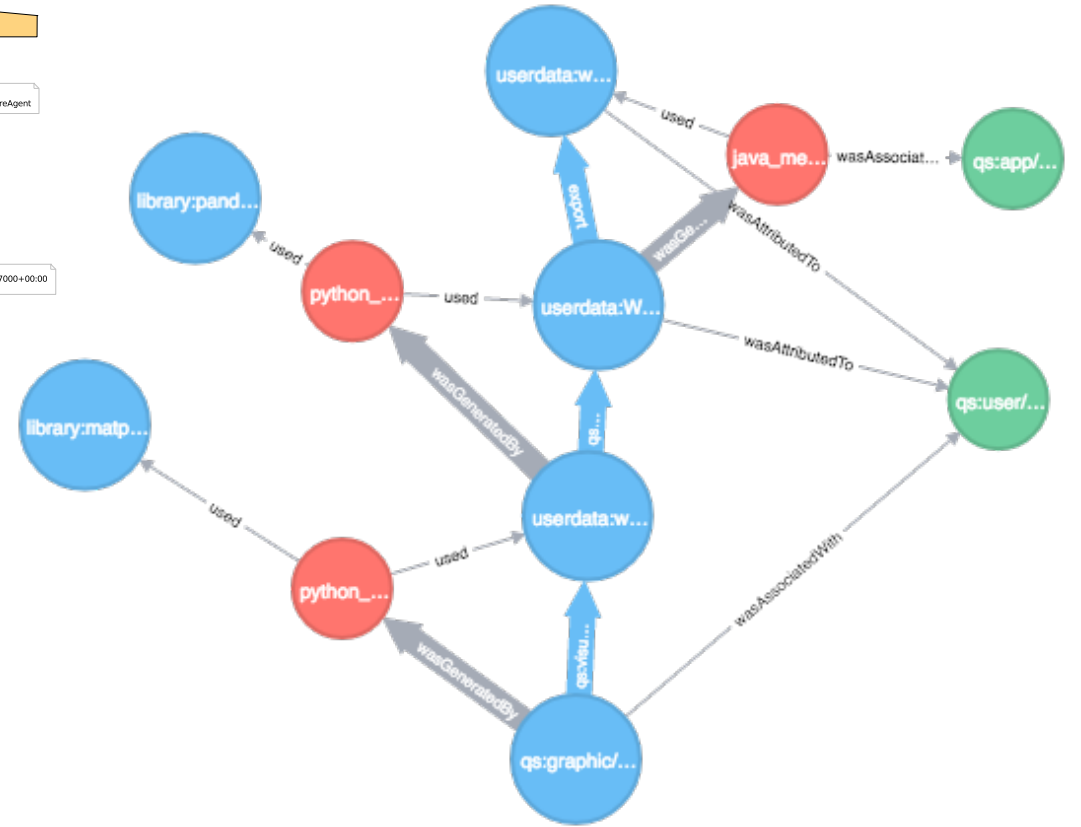
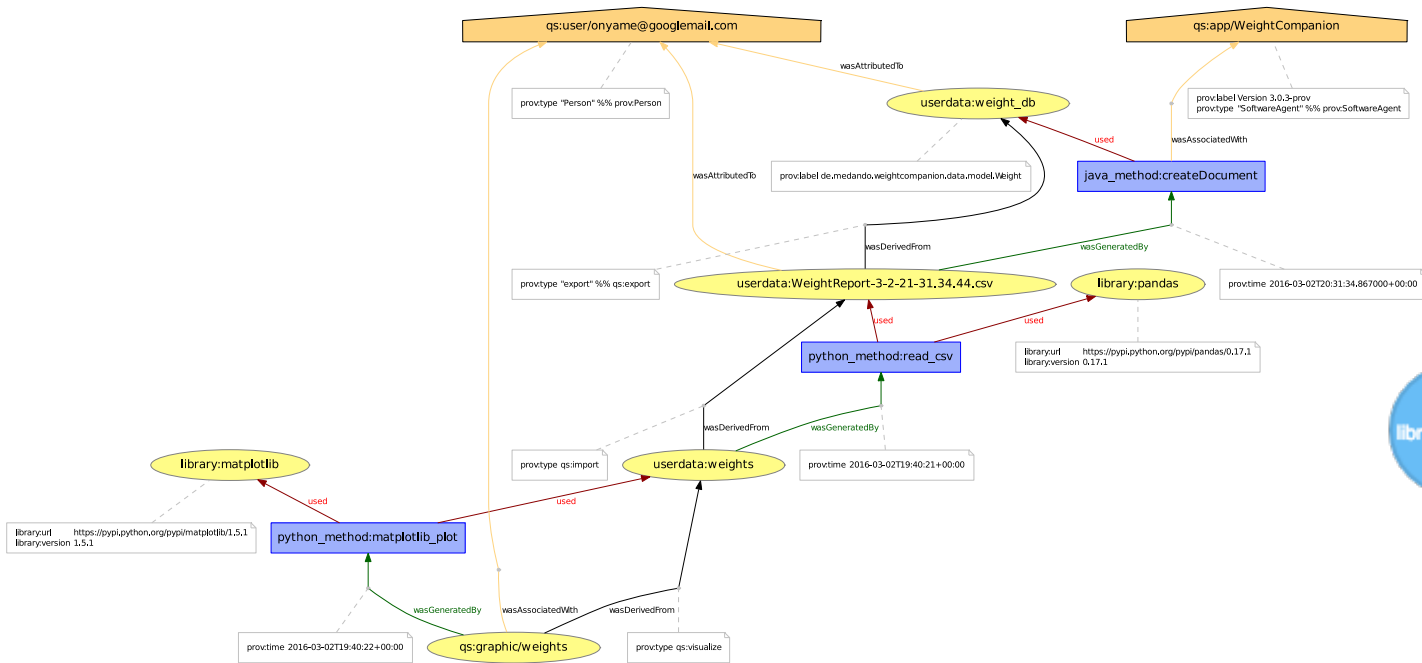
provneo4j_api = provneo4j.api.Api(
    base_url="http://localhost:7474/db/data",
    username="neo4j", password="python")

provneo4j_api.document.create(prov_doc, name="MyProv")
```



PROVNEO4J – Storing PROV Documents in Neo4j

<https://github.com/DLR-SC/provneo4j>



PROV-DB-CONNECTOR

<https://github.com/DLR-SC/prov-db-connector>

Successor of PROVNEO4J

- Connectors for Neo4j implemented
- ArangoDB planned
- APIs in REST implemented
- ZeroMQ & MQTT planned

```
from prov.model import ProvDocument
from provdbconnector import ProvDb
from provdbconnector.db_adapters.in_memory import SimpleInMemoryAdapter

prov_api = ProvDb(adapter=SimpleInMemoryAdapter, auth_info=None)

# create the prov document
prov_document = ProvDocument()
prov_document.add_namespace("ex", "http://example.com")

prov_document.agent("ex:Bob")
prov_document.activity("ex:Alice")

prov_document.association("ex:Alice", "ex:Bob")

document_id = prov_api.save_document(prov_document)

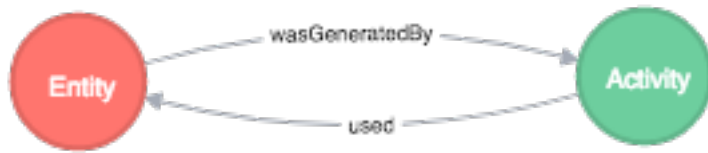
print(prov_api.get_document_as_provn(document_id))
```



Provenance Instrumentation of TensorFlow

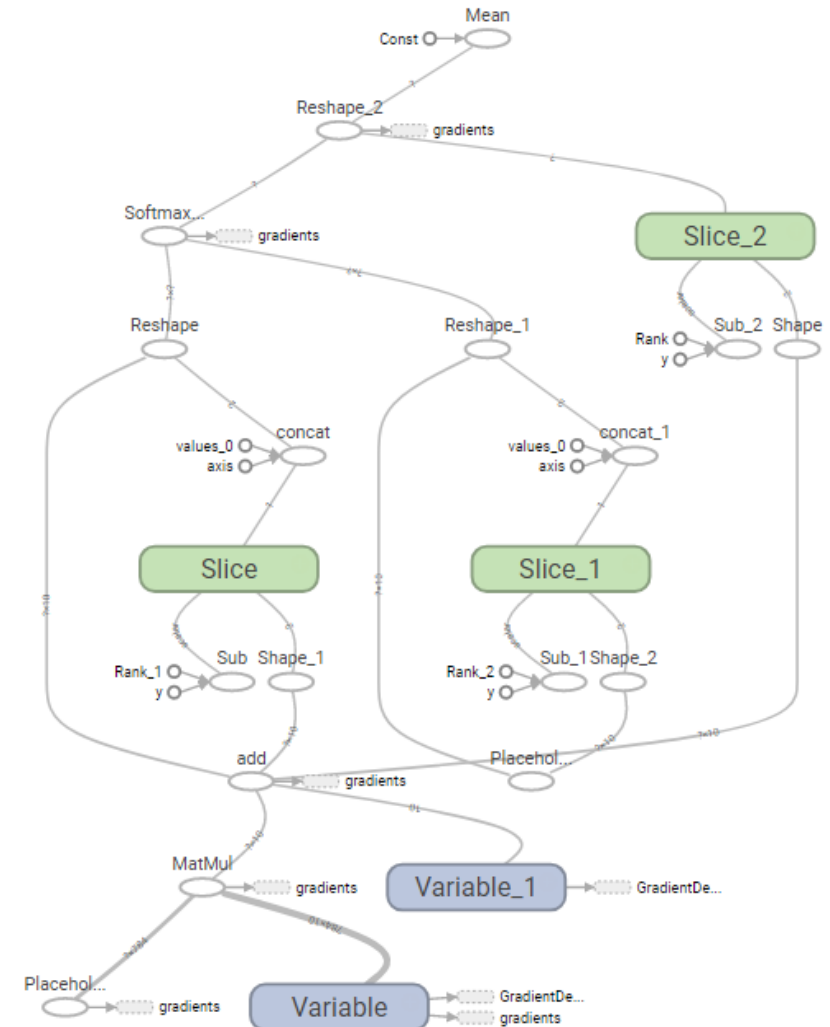
Provenance of TensorFlow workflows

- Tensor → PROV Entity
- Operations → PROV Activity



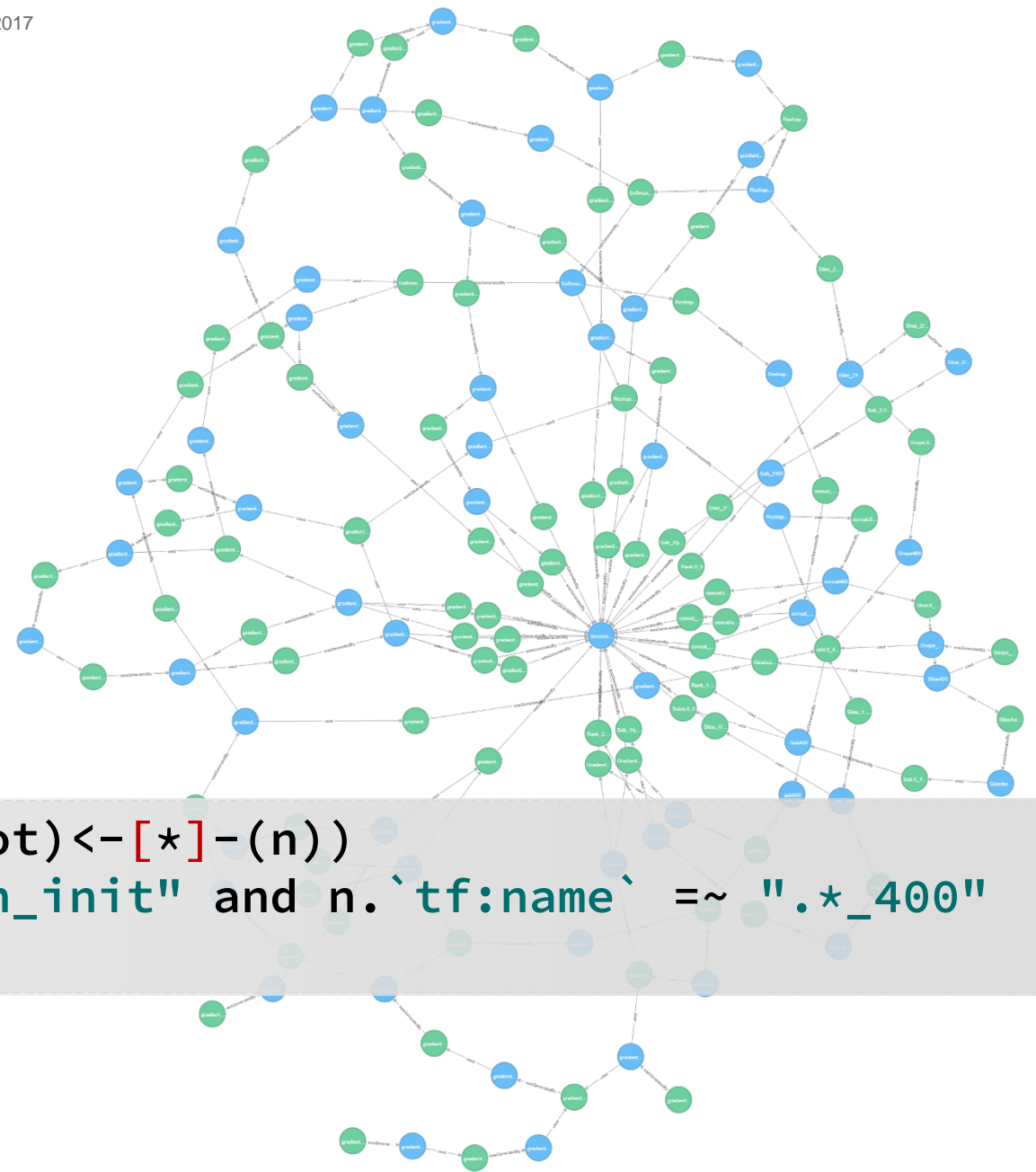
Example: MNIST with 400 training iterations

- 64581 database nodes
- 33549 Entities
- 31032 Activities



Example Query

- Shortest paths from all tensors in 400. iteration to init operation



```
MATCH path=allShortestPaths((root)<-[*]-(n))  
WHERE root.`tf:type`="tf:Session_init" and n.`tf:name` =~ ".*_400"  
RETURN path
```

NoWORKFLOW – Provenance of Scripts

<https://github.com/gems-uff/noworkflow>

```
$ now run -e Tracker experiment.py
```



Project



experiment.py



precipitation.py



p12.dat



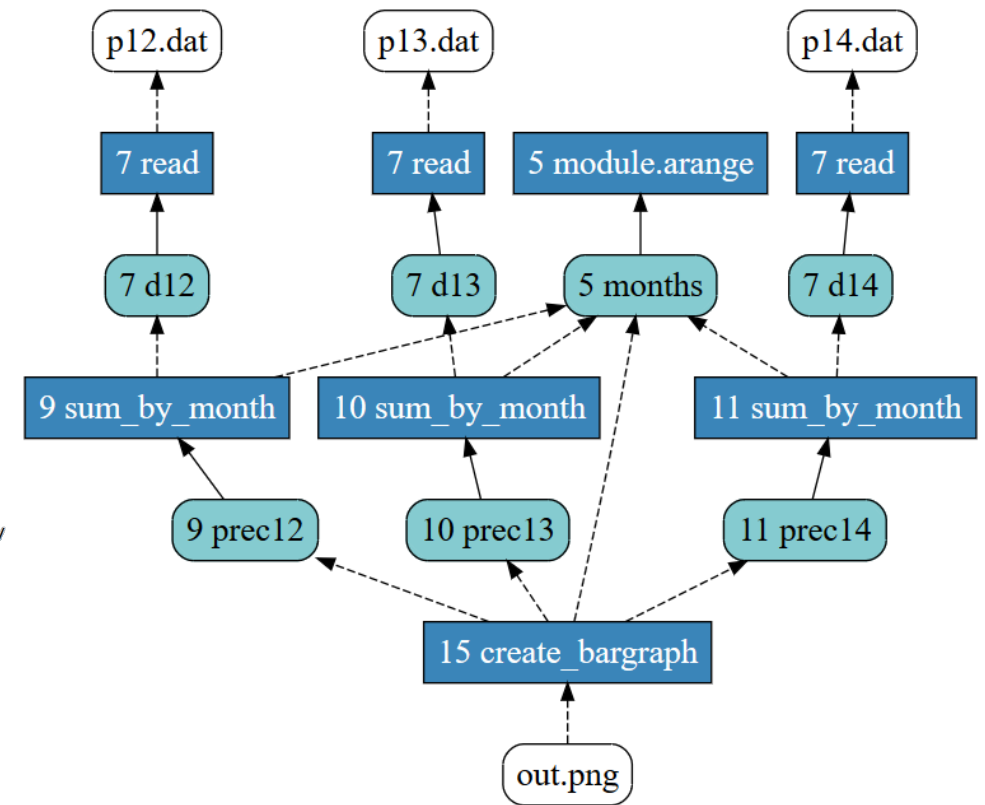
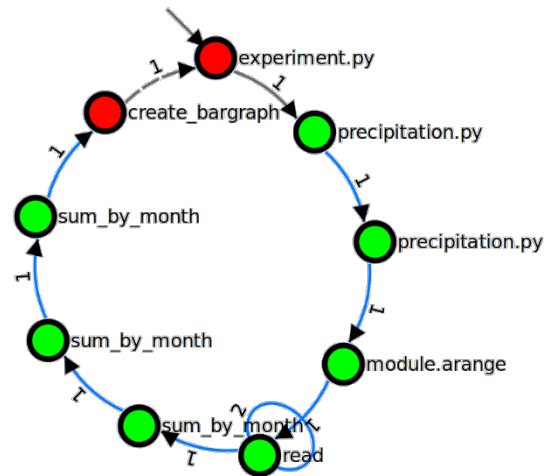
p13.dat



p14.dat



out.png




Git2PROV

<http://git2prov.org>

- Generate PROV documents from git repositories

[Git2PROV](#) [About](#) [Contact](#)



Enter a Git Repo:

Choose a serialization:

PROV-JSON

PROV-N

PROV-O

SVG

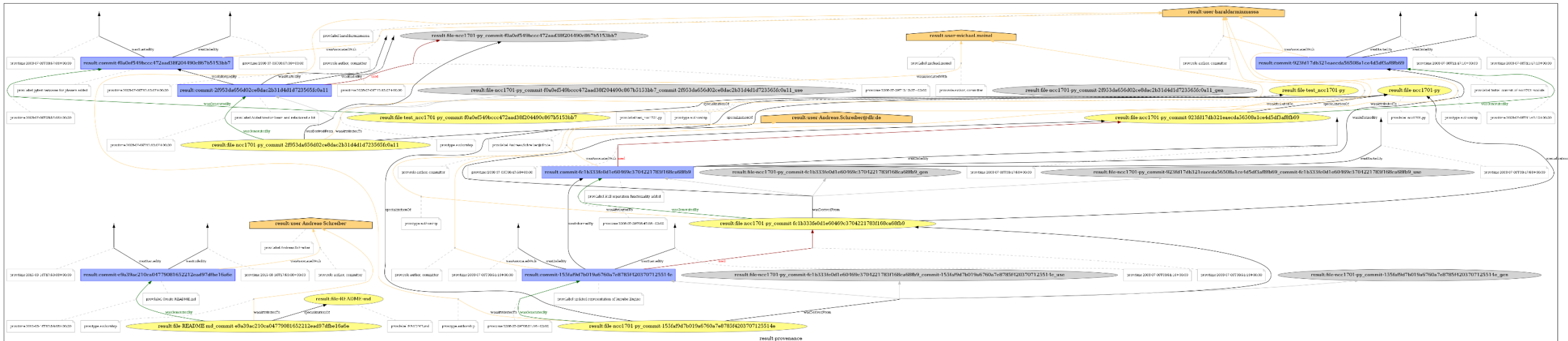
```
{
  "prefix": {
    "result": "http://git2prov.org/git2prov?giturl=https%3A%2F%2Fgithub.com%2FDLR-SC%2Fprovneo4j.git&serialization=PROV-JSON#",
    "fullResult": "http://git2prov.org/git2prov?giturl=https%3A%2F%2Fgithub.com%2FDLR-
```

Download



Git2PROV Example Output

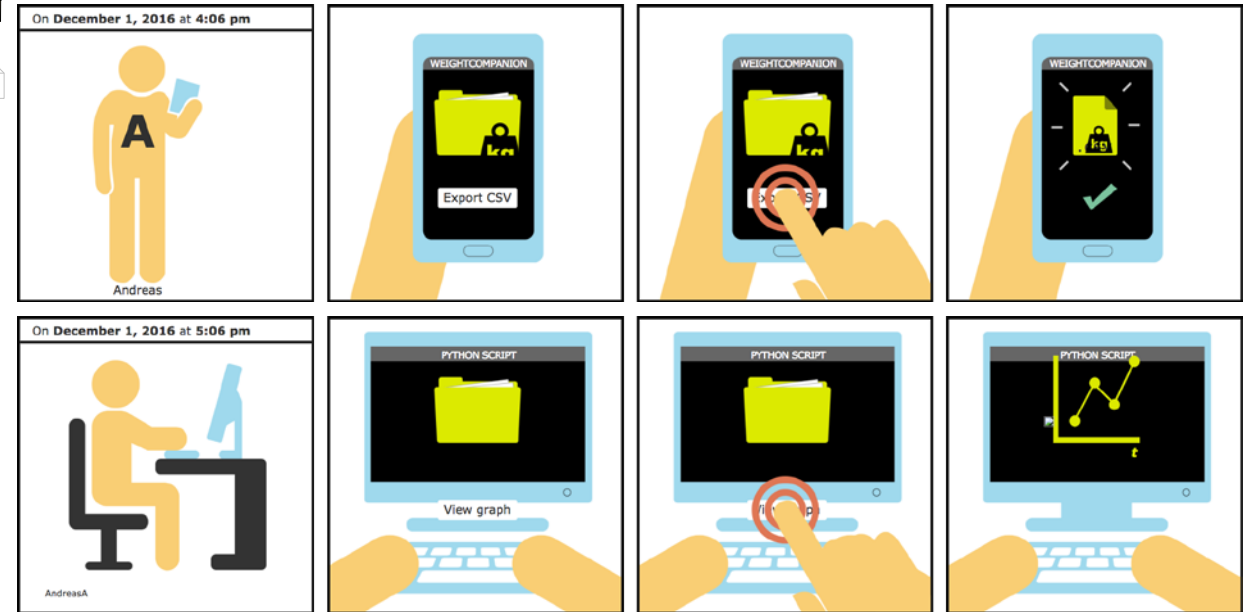
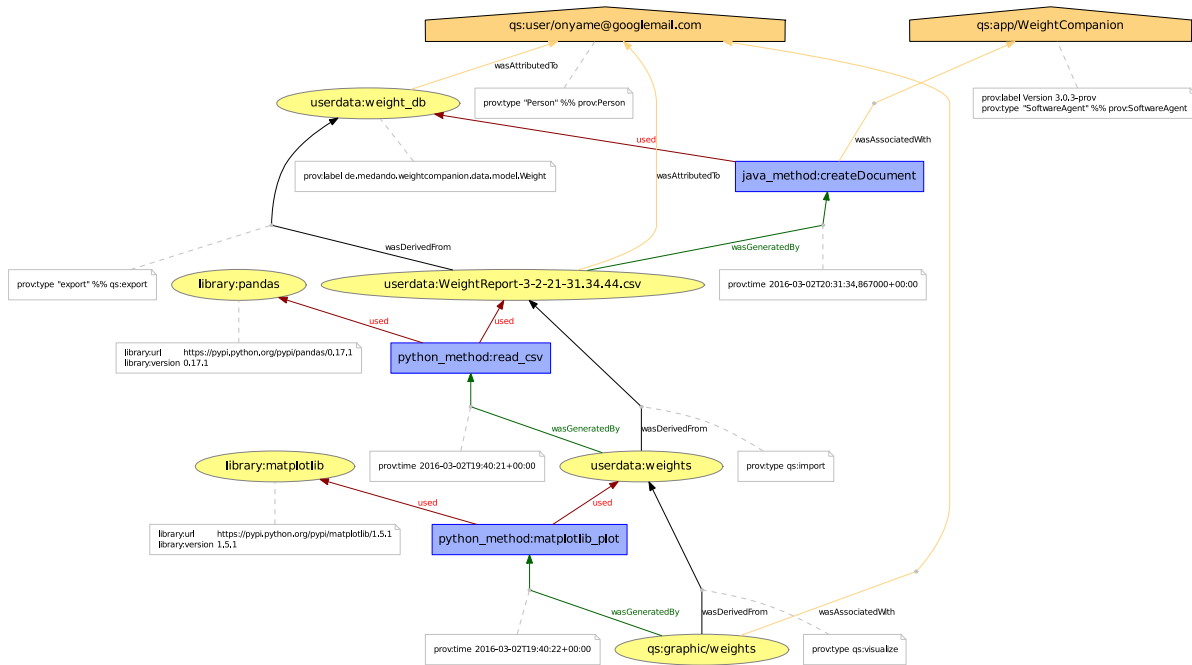
<https://provenance.ecs.soton.ac.uk/store/documents/116377/>



Provenance Visualization

Visualization of Provenance is an ongoing research topic

- Especially, for non-experts (“Provenance for people”)
- Example: PROV COMICS



Key Messages and Summary

Recording the *Provenance* of Data Science workflows is important

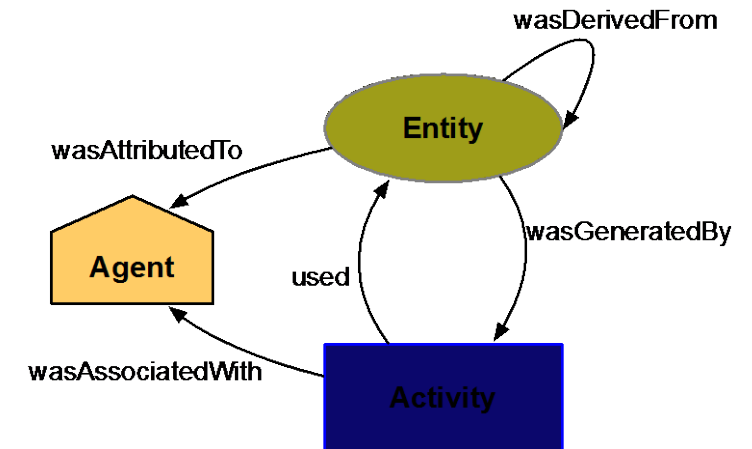
- to understand where data came from
- to reproduce data processing steps or whole workflows

Use a *standard* for Provenance

- W3C standard PROV
- Mapping to (graph) databases, allows easy querying
- A standard allow interoperability and comparison

Recording Provenance is not hard

- APIs for Python
- Tools



Thank You!

Questions?

Andreas.Schreiber@dlr.de
www.DLR.de/sc | @onyame